End-Host Distribution in Application-Layer Multicast: Main Issues and Solutions

Genge Béla and Haller Piroska
Department of Electrical Engineering
"Petru Maior" University of Târgu Mureş
Târgu Mureş, Mureş, Romania, 540088
{bgenge,phaller}@engineering.upm.ro

***Abstract*—Application-layer multicast implements the multicast functionality at the application layer. The main goal of application-layer multicast is to construct and maintain efficient distribution structures between end-hosts. In this paper we focus on the implementation of an application-layer multicast distribution algorithm. We observe that the total time required to measure network latency over TCP is influenced dramatically by the TCP connection time. We argue that end-host distribution is not only influenced by the quality of network links but also by the time required to make connections between nodes. We provide several solutions to decrease the total end-host distribution time.**

Keywords— Multicast; Overlay networks; PlanetLab

I. Introduction

For several years now group communications have been receiving significant attention from both the industry and scientific communities [1], [2]. The main goal of group communication is to enable the exchange of information between group members that can be located across the entire globe.

One of the main application of group communications is in the field of *multicast*. Historically speaking, the first multicast applications were implemented over the IP layer, also known as *IP multicast* [3]. However, after nearly a decade of research in the field of IP multicast, it was never fully adopted because of several technical and administrative issues [4].

Later, there have been several proposals for other multicast implementations that would be easier to deploy over the already existing and well-established Internet protocols and would require little or no modifications in existing routers. Such a survey of existing solutions was provided by El-Sayed et al [5].

One of the directions that has been clearly adopted over the last few years is *application-layer multicast*, which implements the multicast functionality at the application layer. The main goal of application-layer multicast is to construct and maintain efficient distribution structures between *end-hosts*. These structures are constructed using an *overlay* network providing the necessary infrastructure for data transfer between end-hosts.

Today's research focuses on the many aspects of application-layer multicast, including construction of overlay networks [6], [7], optimization issues [8] or security [9]. In our previous work [10] we have addressed the problem of optimally distributing end-hosts (i.e. EH) to overlay network hosts (i.e. OH) in order to minimize network latency and to distribute the load of OH. Based on a heuristic algorithm we proved that the algorithm ensures a local optimal distribution of EH in real time and thus can be used to provide a feasible solution to the distribution problem.

In this paper we focus on the actual deployment of the algorithm proposed in our previous work in a real and globally-scaled distributed system: *PlanetLab* [11]. *PlanetLab* is a "geographically distributed overlay network designed to support the deployment and evaluation of planetary-scale network services" [11]. Using PlanetLab, researchers can test their algorithms and systems in a real environment where nodes can become unreachable, network bandwidth can fluctuate and node processing capabilities can drop dramatically.

In order to test the real applicability of our previously proposed algorithm we have developed an overlay network in PlanetLab where nodes are connected in a complete graph model. There are several advantages for using such a graph model. First, there is no need for implementing complex routing algorithms [12], which greatly simplifies the implementation and functionality of the overlay. Second, maintaining routing tables is not more complex than maintaining connections with all the other nodes. As a downside of this topology, there is a large number of connections that must be maintained, which grows exponentially with the number of OH. However, the simplicity of the routing algorithms between OH makes this topology a great candidate for using it as a leaf component in hierarchical topologies [13], [14].

Existing research [6], [7], [15] focuses on measuring the delay between nodes after the overlay has been constructed or measuring the overlay construction time after TCP connections are done. In deploying our algorithm we have observed that the total time required to measure network latency over TCP is influenced dramatically by the TCP connection time. In this paper we also argue that end-host distribution is not only influenced by the quality of network links but also by the time required to make connections between nodes.

The paper is structured as follows. In Section II we provide an overall presentation of the overlay network, we discuss our previous work and we identify the main problems for deploying the previously proposed algorithm. In Section III we present the measurement results that were done with nodes

spread across 23 countries and we provide 3 solutions for improving the performance of the measurements. Finally, we conclude with an overview of the proposed solutions and we mention some future solutions that could also be implemented.

II. Problem Statement

The measurements that follow in the next sections are based on a complete graph overlay topology where EH are distributed using an heuristic algorithm. An example of such a topology is given in figure Fig. 1, where we have illustrated the presence of 3 host types:

- End-hosts (i.e. EH);
- Overlay-hosts (i.e. OH);
- Monitor-hosts (i.e. MH).

EH are the producers and consumers of data transferred by the overlay containing the OH. MH are used to monitor the load of each OH and to distribute the connection of EH. The heuristic algorithm we proposed in our previous work is used to distribute EH to OH in order to minimize latency and to distribute the load of OH.

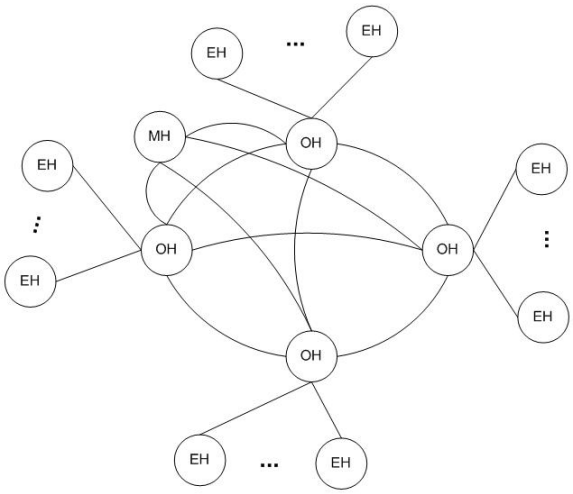

Fig. 1. Multicast topology

The distribution algorithm uses the measured latency between all OH pairs, the load of each OH and the measured latency between each EH and OH pairs. The algorithm is run by the MH each time a new EH must be connected. At this time, the EH must provide the MH its measurement results on the network latency it recorded to each OH. Based on this data and the reported load received from each OH, the MH runs the distribution algorithm.

As mentioned in our previous work, after all data is available, the algorithm executes very fast. For instance, from the simulations we run, for 100 OH the algorithm execution time for distributing a single EH is about 3.7 ms. This execution time provides a real-time applicability of the proposed algorithm.

We have chosen to deploy the proposed multicast in PlanetLab because it provides globally-available network services that can be used to run any application type that can run on a Linux OS. From the beginning of the implementation process we had to deal with several problems. First of all, network connections between PlanetLab nodes or even node CPUs can be heavily loaded, sometimes even leading to SYN_ACK timeouts for TCP connections. Second, nodes can be rebooted at anytime by PlanetLab Central coordinators in order to ensure a software update, for software maintenance or simply because of some hardware problems. These problems must be handled by the MH in order to ensure that EH are not distributed to such nodes and that already distributed EH nodes are redistributed if necessary (i.e. on OH failure).

We also encountered several problems on the EH side. The proposed algorithm heavily relies on the measurement data provided by EH. This means that when joining the network, all EH must first measure the latency with all OH and then send this data to MH. The problem with this approach is that in some cases the response time from OH is very long, in the order of seconds as shown in the next sections. This leads to an overall distribution time in the order of seconds or even minutes, which is unacceptable.

TABLE I
Country and OH node count

Country	Node count	Country	Node count
Austria	1	Italy	6
Canada	2	Korea	2
France	4	Poland	3
Germany	9	Romania	2
Greece	1	Spain	2
Hungary	1	Switzerland	1
Israel	1	US	5

III. Measurement Issues and Solutions

A. Overlay Construction Time

Although the construction of the overlay is done only once, we consider that measuring the construction time can provide useful perspective of the time required to re-construct the overlay in possible future developments. The constructing of the overlay network is not made instantly. In order to evaluate the performance and the general usability of the proposed overlay, we have measured the time needed to construct the complete graph between the overlay nodes.

Deploying and starting applications on PlanetLab nodes can be done automatically using applications such as *multicopy* or *multiquery* that are part of the CoDeploy project [16]. These allow a parallel deployment and execution of commands on a set of nodes. We have considered 5 settings with a different number of OH nodes. The OH applications were deployed on nodes from 14 countries (for the maximum number of 40 OH nodes), as shown in Table I. After starting the OH applications, each OH connects to all other OH according to Alg. 1, where OH corresponds to the set of OH, Cout is the set of outgoing connections and Cin is the set of incoming connections.

At first, each OH starts the connection process to other OH nodes. Then, it waits for the connection process to complete.

Algorithm 1 Complete connections for one OH

```
Let t1 = @Get_curr_time()
Let Cout = φ

{Start connection sequences}
for all oh ∈ OH do
  c = @Start_conn_sequence(oh)
  Cout = Cout ∪ {c}
end for

{Wait for completion}
@Wait_for_completion( Cout )

{Now eliminate duplicate connections}
Let Cin = @Get_incoming_connections()
for all c ∈ Cout do
  if ∃c' ∈ Cin :@Src_address(c')=@Dest_address(c) then
    (Meas_out, Meas_in)=@Run_measurements(c, c')
    if Meas_out < Meas_in then
      @End_connection(c)
      Cout = Cout \ {c}
    end if
  end if
end for

{Calculate complete connection time}
Let t2 = @Get_curr_time()
Let G_Time = t2 − t1
```

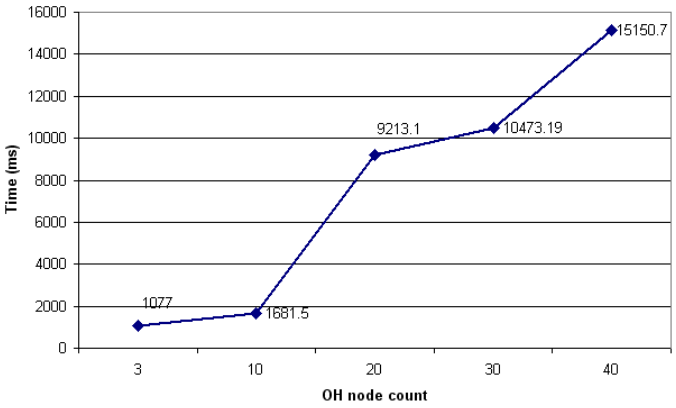

Fig. 2. Complete graph construction time

This process leads to duplicate connections between each OH node pair. In order to eliminate duplicate connections we measure the connection latency in each direction by sending a single package of 1500Bytes and we eliminate the connection with the maximum latency.

According to Alg. 1, each OH calculates a complete connection time G_{Time}. The complete graph construction time is the maximum of these values, as shown in Fig. 2. As we can see in Fig. 2 the construction of the overlay is greatly influenced by the number of nodes. However, the variation is not linear because the overlay also depends on other factors such as the quality of network connections and the load of nodes. The result shown in Fig. 2 has the following explanation. In the first OH set (i.e. 3 nodes), all 3 nodes are located in European countries, with a minimum load. In the next OH set (i.e. 10 nodes) we have added additional nodes from Europe, one node from the US and one node from Asia. This almost doubled the graph construction time because the node from Asia was heavily loaded, with the CPU running at over 80% almost all the time. In the next set (i.e. 20 nodes) we have added additional nodes from Asia, Canada and Europe which, because of network connection latencies and heavily loaded nodes (i.e. from Israel and Germany) has led to a quadruple time. In the next two sets (i.e. 30 and 40 nodes) we have added additional nodes from Europe and US, leading to the results shown in Fig. 2.

B. EH Connection Measurement Issues

When EH nodes are started, each node first connects to all OH nodes in order to measure the network latency. The measured values are then sent to the MH that applies the heuristic algorithm developed in our previous work [10] to determine the OH node where each EH must connect. We have identified two components that significantly influence the measured values: connection time and network latency. Let EH be the set of EH. Then, the total measurement time M_i needed to be executed by an EH is:

$$M_i = \max_{oh_j}\{Conn(eh_i, oh_j) + CummLat(eh_i, oh_j)\} \quad (1)$$

where $eh_i \in \mathsf{EH}$, $i = \overline{1, |\mathsf{EH}|}$ and $oh_j \in \mathsf{OH}$, $j = \overline{1, |\mathsf{OH}|}$. $Conn$ denotes the time needed to establish a connection between eh_i and oh_j. $CummLat$ denotes the cumulated round-trip latency calculated by measuring the time difference between sent and received packages:

$$\begin{aligned} CummLat(eh_i, oh_j) \quad = \quad & Lat_1(eh_i, oh_j) + \\ & Lat_2(eh_i, oh_j) + \\ & Lat_3(eh_i, oh_j) \end{aligned} \quad (2)$$

where Lat_1, Lat_2 and Lat_3 denote the round-trip latency of 3 packages.

We have considered several scenarios, with EH count ranging from 10 to 1000. EH nodes were deployed on nodes from 23 countries (for the maximum number of 1000 EH nodes), as shown in Table II.

Each EH calculates its own M_i value that is sent to the MH that calculates an average measurement time, illustrated in Fig. 3. We can see that the number of OH nodes clearly influences the overall measurement time. There are several values that break the linear trajectory. For instance, in the case of 40 OH nodes, when running 50 EH nodes the average time is 39382ms and when running 100 EH nodes the average time is reduced to 21571ms. The explanation for this behavior lies in

TABLE II
COUNTRY AND EH NODE COUNT

Country	Node count	Country	Node count
Argentina	10	Japan	10
Australia	10	Korea	20
Austria	40	Netherlands	20
Belgium	20	Poland	40
Canada	100	Portugal	10
China	20	Romania	20
Finland	10	Russia	20
France	110	Spain	40
Germany	160	Switzerland	10
Greece	10	Taiwan	10
Hungary	20	US	240
Italy	60		

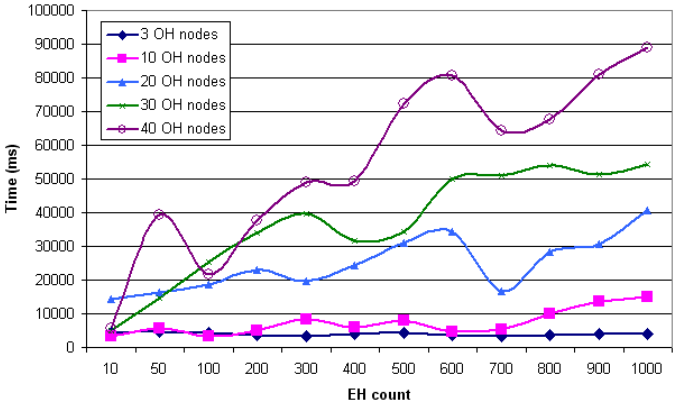

Fig. 3. Average EH measurement time

the way that the measurements were done. Because PlanetLab offers a set of resources over the Internet that are shared among researchers, time measurements can change dramatically from one execution to another. Moreover, the measurements we made span across 10 days. We have actually seen that in one day a given node can be extremely loaded because other researchers may also be running experiments, and the next day the node can show a minimum load. This is in fact the expected behavior of nodes running in a real networking environment that greatly differs from the controlled laboratory environments.

The values shown in Fig. 3 include both the connection time and the network latency. However, as we can see from Fig. 4, the latency is only a small part of the measurement time, with average values ranging from 68.59ms to 925.86ms.

The values shown in Fig. 3 clearly show that we should improve the performance of the measuring algorithm. At this stage, the average time needed to measure the network latency for 1000 EH nodes in the 40 OH node setting is 89000ms, which corresponds to almost 1.5 minutes. However, this is the average time, which is much smaller than the maximum time needed for an EH to make the measurements. The maximum measurement time is shown in Fig. 5, where we can see that the maximum time needed to make the measurements is in fact 561192ms, which is almost 9.5 minutes. The values from Fig. 5 show that the time needed for all nodes to make the

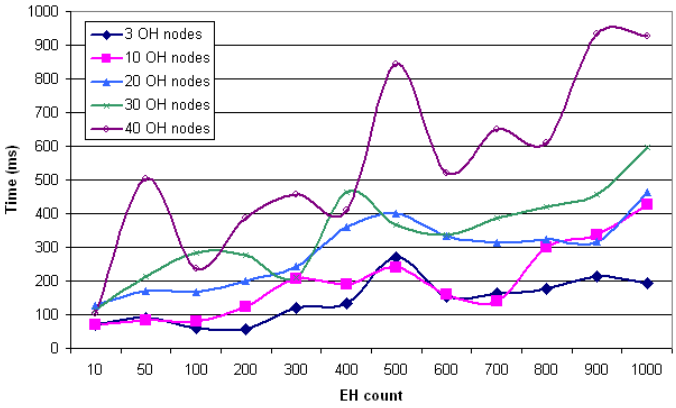

Fig. 4. Average EH-OH measured latency

measurements are influenced by the number of OH and by the number of EH, leading to the value of 9.5 minutes, which is unacceptable.

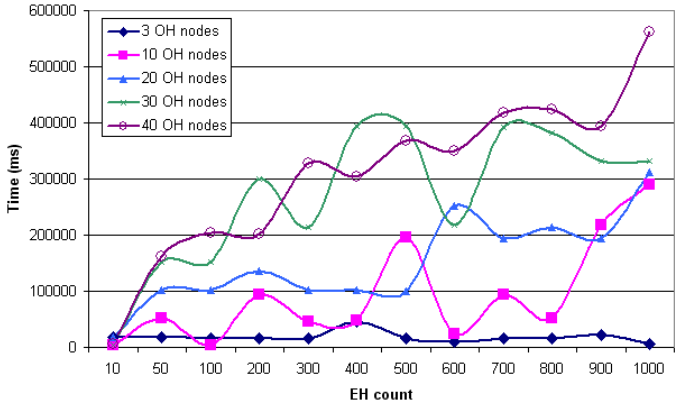

Fig. 5. Maximum EH measurement time

The total accessing distribution time of EH is also influenced by the response time from the MH. In all our measurements the MH resides on a single node from Romania. In Fig. 6 we can see the average response time from the MH. Interestingly, the response time is not influenced by the number of OH or by the number of EH, but by the number of simultaneous requests that are received. EH nodes connect to MH only after completing the measurements, this is why when a large number of EH connect simultaneously to the MH we get the peaks from the figure. From the measurements we have also seen that after receiving the measurement data the distribution algorithm is running under 1ms for each request, thus the values shown in Fig. 6 are given by message processing and network delay.

After an EH successfully connects to the OH, it can stay connected for an unlimited time. However, if the connection is interrupted, it will reconnect to the designated OH. If the designated OH is no longer available, it must execute the measurement and distribution all over again. In case of new EH nodes, these are distributed by the MH without redistributing

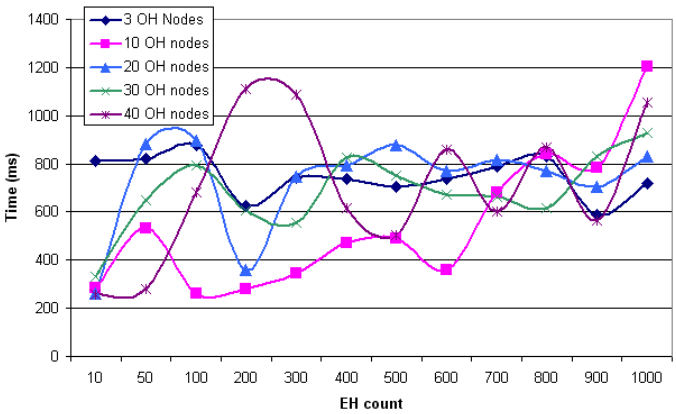

Fig. 6. Average MH response time

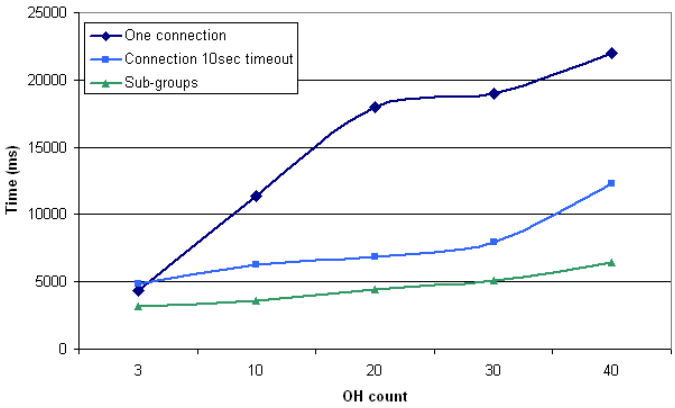

Fig. 7. Average improved EH measurement time for 1000 EH

the already connected EH nodes.

As mentioned earlier, in case of OH failure, disconnected EH nodes initiate a new measurement and distribution process. However, in case of network failures between OH nodes, a reconnect mechanism is activated for each OH node that tries to re-establish connection with all other OH nodes, effectively trying to reconstruct the overlay.

C. EH Connection Measurement Solutions

As illustrated in the previous section, making network measurements at the application layer is mainly influenced by the connection time between nodes. The network latency factor, as opposed to the connection time, has a minimum impact on the total time.

When EH use the proposed overlay, their main goal is not to make measurements but to actually use it to effectively distribute data. The time needed to make the measurements should thus be reduced to a minimum possible.

In this section we propose 3 solutions to the measurement problem. After implementing them, we have repeated the measurements for the 1000 EH setup, where the modifications would have a greater impact.

The first solution involves reducing the reconnect process count to 0, meaning that if a connect attempt fails, the EH removes the OH from its list. EH nodes usually try to connect over and over again to OH nodes until successful. This process dramatically increases the overall measurement time, as shown in the previous section. By eliminating the reconnections, we are in fact eliminating OH that are overloaded or to which we have a poor connection. The improvements can be immediately seen, as shown in Fig. 7. In this case, for the maximum setting, with 40 OH nodes, the average measurement time drops from 89000ms to 22027ms, improving the overall measurement 4 times.

The problem with the first solution is that a connection must be timed out by the OS to eliminate the OH from the solution. As a second solution we propose an application-controlled connection timeout, opposed to network OS timeout. In this case we timed out connections that exceeded 10 seconds, decreasing the average measurement time from 89000ms to 12284ms and improving the overall measurement 7 times, as shown in Fig. 7. The 10 seconds were chosen based on the observation that a lower timeout leads to an increased number of OH nodes eliminated from the solution. This problem is discussed in detail later in this section.

TABLE III
SUB-GROUP PARTITIONING

Sub-Group	**3 OH** 1OH/EH	**10 OH** 2OH/EH	**20 OH** 4OH/EH	**30 OH** 6OH/EH	**40 OH** 8OH/EH
Grp1	333 EH	200 EH	200 EH	200 EH	200 EH
Grp2	333 EH	200 EH	200 EH	200 EH	200 EH
Grp3	333 EH	200 EH	200 EH	200 EH	200 EH
Grp4	-	200 EH	200 EH	200 EH	200 EH
Grp5	-	200 EH	200 EH	200 EH	200 EH

The third solution involves partitioning the OH and EH nodes into sub-groups, thus reducing the total number of OH/EH and the total number of EH/OH. The partitioning can be seen in Table III. As we can see from Fig. 7, the average time required for measurements is reduced to 6459ms for 40 OH nodes, improving the overall measurement time over 13 times.

The direct effect of the first two solutions is that the number of OH nodes for which EH nodes test the connection reduces significantly with the reduction of the timeout. For instance, by using the OS timeout, which can range from a few seconds to a few minutes we have less eliminated OH nodes than using a fixed timeout of 10 seconds, as shown in Fig. 8. In case of only one connection (i.e. OS timeout) the tested percentage is 100% for 3 OH nodes, however, this drops to 95% for 10 and 20 nodes and then rises to 96.66% for 30 nodes and to 97.43% for 40 nodes. In case of application-layer timeout we have a 98.1% for 3 OH nodes which drops to 71.79% for 40 OH nodes.

Although the partitioning-based solution provides the best timings, it can limit sub-groups to a set of OH nodes that may not provide the optimal solution for the entire group. While the application-layer timeout mechanism seems to be the next best approach, care must be taken in choosing the timeout

value because a larger connection-time does not necessarily mean that the specific node is heavily loaded, but several other factors can also influence this value, such as a momentarily busy OS, or a momentarily busy application.

Other solutions could also be applied, such as using UDP for determining the network latency between EH and OH. Such a solution would eliminate the overhead given by TCP connection. However, because the overlay uses TCP for forwarding data, making measurements by connecting to OH nodes via TCP provides a more precise view on the future behavior of OH nodes.

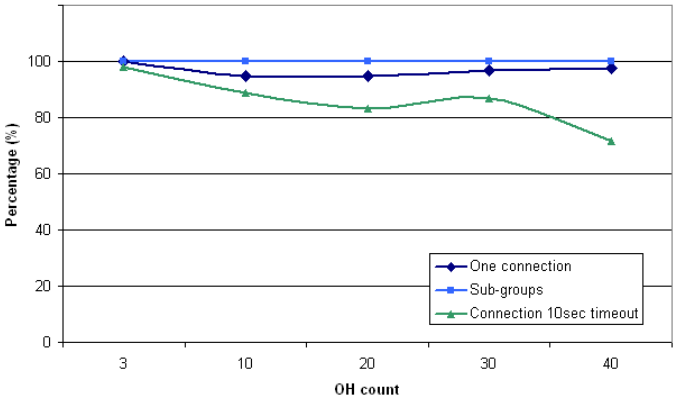

Fig. 8. Average percentage of measured connections

IV. Conclusions and Future Work

We presented several issues and solutions for deploying application-layer overlay networks. Based on our measurements conducted over PlanetLab, a real network testing platform, we have concluded that distributing EH nodes can not be based only on the measured network latency, but must also include other elements such as connection time or EH geographical location to reduce the time required to make the actual latency measurements.

The identified problems have several solutions. In this paper we have proposed 3 such solutions: a first one that eliminates reconnections, a second one that uses application-layer timeouts and a third one that constructs sub-groups for reducing the number of OH/EH and EH/OH. By using these solutions we have shown that the measurement time can be reduced up to 13 times for 1000 EH and 40 OH.

As future work, we intend to use UDP for the initial measurements. However, special care must be taken because a lower timing for UDP packages does not necessarily imply lower timings for TCP packages. A study must be made to determine the correspondence between UDP and TCP timings and how could UDP-based measurements be used to forecast the overhead introduced by TCP connections. This study must also take into consideration UDP packet losses that may also influence the total measurement time.

References

[1] F. Bacelli, A. Chaintreau, Z. Liu, A. Riabov, and S. Sahu, "Scalability of Reliable Group Communication Using Overlays", Proceedings of INFOCOMM, 2004.

[2] S. M. Venilla, and V. Sankaranarayanan, "Threat Analysis for P-LeaSel, a Multicast Group Communication Model", Asian Journa of Information Technology, Vol. 7, 2008, pp. 64–68.

[3] S. Deering, and D. Cheriton, "Multicast Routing in Datagram Internetworks and Extended LANS", ACM Transactions on Computer Systems, Vol. 8, No. 2, 1990, pp. 85–111.

[4] C. Diot, B.N. Levine, B. Lyles, H. Kassem, and D. Balensiefen, "Deployment issues for the IP multicast service and architecture", IEEE Network Magazine, Vol. 14, No. 1, 2000, pp. 78–88.

[5] A. El-Sayed, and V. Roca, "A Survey of Proposals for an Alternative Group Communication Service", IEEE Network, Vol. 17, No. 1, 2003, pp. 46–51.

[6] V. Roca, and A. El-Sayed, "A Host-Based Multicast (HBM) Solution for Group Communications", Proceedings of the First International Conference on Networking, LNCS, Vol. 2093, 2001, pp. 610–619.

[7] K. Ragab, and A. Yonezawa, "A Self-organized Clustering-based Overlay Network for Application Level Multicast", Journal of Networks, Vol. 4, No. 2, 2009, pp. 85–91.

[8] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen, "Polynomial Time Algorithms for Multicast Network Code Construction", IEEE Transactions on Information Theory, Vol. 51, No. 6, 2005, pp. 1973–1982.

[9] N. Shanthi, and L. Ganesan, "Security In Multicast Mobile Ad-Hoc Networks", International Journal of Computer Science and Network Security, Vol. 8, No. 7, 2008, pp. 326–330.

[10] H. Piroska, and R. Balint, "Optimal server distribution in multimedia communication", IN the Proc. of the 4th International Conference on RoEduNet, 2005, pp. 142–147.

[11] A. Bavier, M. Bowman, B. Chun, D. Culler, S. Karlin, S. Muir, L. Peterson, T. Roscoe, T. Spalink, and M. Wawrzoniak, "Operating System Support for Planetary-Scale Network Services", Networked Systems Design and Implementation, 2004.

[12] T. L. Huang, and D. T. Lee, "A distributed multicast routing algorithm for real-time applications in wide area networks", Journal of Parallel and Distributed Computing, Vol. 67, Issue 5, 2007, pp. 516–530.

[13] W. Jia, W. Tu, and J. Wu, "Hierarchical Multicast Tree Algorithms for Application Layer Mesh Networks", Networking and Mobile Computing, LNCS, Vol. 3619, 2005, pp. 549–559.

[14] W. Yong, W. Seng, and H. Xianying, "A new Hierarchical Application Layer Multicast algorithm for large-scale video broadcasting", In the Proc. of the 2nd IEEE International Conference on Computer Science and Information Technology, 2009, pp.610–613.

[15] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Application-Level Multicast Using Content-Addressable Networks", Networked Group Communication, LNCS, Vol. 2233, 2001, pp. 14–29.

[16] K. Park, and V. Pai, "Deploying Large File Transfer on an HTTP Content Distribution Network", In Proceedings of the First Workshop on Real, Large Distributed Systems (WORLDS '04), 2004.